**POLYGLOT – Chatbot for Simple Question and Answers**

**1. User Input Handling**

* The user types a message in the input box.
* The message can be in English or any supported Indian language.
* Once the message is sent, it appears in the chat UI on the right side under the label **"You:"** with a timestamp.
* The chatbot's response appears on the left side under the label **"Bot:"**, also with a timestamp.

**2. Message Processing Workflow**

1. The user's input is captured and sent to the function **send\_message\_backend()**.
2. A "Bot is typing..." animation appears while the system processes the response.
3. The chatbot first checks for a **predefined response** in the pretrained\_responses dictionary:
   * If a matching response exists, it is returned instantly.
   * If no predefined response is found, the message is sent to **Google Gemini AI (Gemini API)** to generate a response.
4. The generated response is then translated into the user's selected language using **Google Translate API**.
5. The translated response is converted into speech using **Google Text-to-Speech (gTTS)**.
6. The final text response, along with an embedded audio player, is displayed in the chat.

**3. Chat Display & Formatting**

* **User Messages:**
  + Appears in a green background with right alignment.
  + Includes a timestamp below the message.
* **Bot Messages:**
  + Appears in a dark gray background with left alignment.
  + The translated text is displayed along with a playable audio response.
* **Auto-Scrolling Feature:**
  + Ensures that the latest messages are always visible without manual scrolling.

**4. Example Conversation Output**

**User:** "Hello!" (Sent in English)

* **Bot:** "Hey there! What can I do for you?" (Displayed in English with an audio option)

**User:** "आप कैसे हैं?" (Sent in Hindi)

* **Bot:** "मैं कोड हूं, लेकिन पूछने के लिए धन्यवाद! " (Displayed in Hindi with an audio option)

**User:** "What is AI?" (Sent in English)

* **Bot:** "Artificial Intelligence (AI) is the simulation of human intelligence in machines..." (Generated by AI, translated, and spoken)

**Note:** Coppy the code and run it in a “**colab-notebook**”

**CODE**

**# --- Install Required Packages ---**

!pip install -q --upgrade gTTS googletrans==4.0.0-rc1 httpx>=0.28.1

**# --- Imports Libraries ---**

import google.generativeai as genai

import ipywidgets as widgets

from IPython.display import display, HTML, Javascript

from gtts import gTTS

from googletrans import Translator

import base64

import re

from datetime import datetime

**# --- API & Model ---**

genai.configure(api\_key="AIzaSyAESbu-jTTS3j9rR\_Yo58WMZmdhE4Yz8j0")

model = genai.GenerativeModel("gemini-1.5-pro")

translator = Translator()

**# --- Pretrained Response Map ---**

pretrained\_responses = {

    "hi": "Hello! How can I assist you today?",

    "hello": "Hey there! What can I do for you?",

    "hey": "Hi! Need any help?",

    "how are you": "I'm just code, but thanks for asking! 😊",

    "help": "Sure! You can ask me questions in multiple Indian languages, and I’ll respond with text and voice.",

    "what's up": "Just waiting to assist you!",

    "good morning": "Good morning! Hope you have a great day!",

    "good night": "Good night! Sleep well!",

    "bye": "Goodbye! Take care!",

}

**# --- Core Logic Functions ---**

def get\_ai\_response(user\_input):

    try:

        response = model.generate\_content(user\_input)

        return response.text

    except Exception:

        return "⚠️ Error: Unable to process request."

def translate\_text(text, target\_lang="en"):

    try:

        translated = translator.translate(text, dest=target\_lang)

        return translated.text

    except Exception:

        return "⚠️ Error: Translation failed."

def clean\_text(text):

    return re.sub(r"[^\w\s.,!?]", "", text)

def generate\_audio\_base64(text, lang="en"):

    try:

        sanitized\_text = clean\_text(text)

        tts = gTTS(text=sanitized\_text, lang=lang, slow=False)

        audio\_path = "response\_audio.mp3"

        tts.save(audio\_path)

        with open(audio\_path, "rb") as audio\_file:

            audio\_base64 = base64.b64encode(audio\_file.read()).decode("utf-8")

        return f"""

        <audio controls style="width:100%; margin-top:5px; border-radius:5px; background:#222; height:30px;">

            <source src='data:audio/mp3;base64,{audio\_base64}' type='audio/mp3'>

        </audio>

        """

    except Exception:

        return "⚠️ Error: Unable to generate audio."

**# --- Styling ---**

bootstrap\_style = """

<style>

    body {

        background-color: #1c1c1c !important;

        color: white;

    }

    #container {

        display: flex;

        flex-direction: column;

        align-items: center;

        margin-top: 20px;

    }

    #chat-container {

        background-color: black;

        padding: 15px;

        padding-left:30px;

        border-radius: 12px;

        width: 347px;

        height: 600px;

        position: fixed;

    }

    #chat {

        width: 350px;

        height: 500px;

        overflow-y: auto;

        padding: 10px;

        margin:10px 10px 10px 8px;

        border-radius: 12px;

        background: linear-gradient(to bottom right, #2e2e2e, #1f1f1f);

        box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.5);

        display: flex;

        flex-direction: column;

        justify-content: flex-start;

        scrollbar-width: none;

    }

    #chat::-webkit-scrollbar {

        width: 0px;

        background: transparent;

    }

    .message {

        max-width: 80%;

        padding: 10px;

        border-radius: 12px;

        font-size: 14px;

        word-wrap: break-word;

        margin-bottom: 10px;

        display: inline-block;

        clear: both;

        position: relative;

    }

    .user-message {

        background: #4CAF50;

        color: white;

        align-self: flex-end;

        text-align: right;

        border-bottom-right-radius: 5px;

        margin-left: auto;

    }

    .bot-message {

        background: #333;

        color: white;

        align-self: flex-start;

        text-align: left;

        border-bottom-left-radius: 5px;

        margin-right: auto;

        box-shadow: 0px 2px 4px rgba(0, 0, 0, 0.3);

    }

    .timestamp {

        font-size: 10px;

        color: #aaa;

        margin-top: 2px;

        display: inline-block;

    }

    .icon {

        font-size: 16px;

        margin-right: 5px;

    }

    audio {

        width: 100%;

        border-radius: 5px;

        background: #2c2c2c;

    }

    #chat-input-wrapper {

        margin:10px 8px 10px 8px;

        width: 350px;

        background: #1f1f1f;

        padding: 10px;

        border-radius: 10px;

        box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.3);

        position:sticky;

    }

    .input-group {

        display: flex;

        flex-direction: row;

        gap: 10px;

        justify-content: center;

        align-items: center;

        width: 100%;

    }

    .form-select, .form-control, .btn {

        padding: 8px;

        font-size: 14px;

        border-radius: 6px;

        border: none;

        outline: none;

    }

    .form-select {

        background-color: #2b2b2b;

        color: white;

        width: 25%;

        text-align: center;

    }

    .form-control {

        background-color: #2b2b2b;

        color: white;

        width: 50%;

    }

    .btn {

        background-color: #4CAF50;

        color: white;

        cursor: pointer;

        width: 20%;

    }

    .btn:hover {

        background-color: #45a049;

    }

    /\* Typing animation \*/

    @keyframes bounce {

        0%, 80%, 100% { transform: scale(1); }

        40% { transform: scale(1.3); }

    }

    .typing-dots span {

        display: inline-block;

        animation: bounce 1.2s infinite;

        font-weight: bold;

    }

    .typing-dots span:nth-child(2) { animation-delay: 0.2s; }

    .typing-dots span:nth-child(3) { animation-delay: 0.4s; }

</style>

"""

auto\_scroll\_js = """

<script>

function autoScroll() {

  var chatDiv = document.getElementById('chat');

  chatDiv.scrollTop = chatDiv.scrollHeight;

}

</script>

"""

**# --- Input UI ---**

input\_ui\_html = """

<div id="chat-input-wrapper">

    <div class="input-group">

        <select id="langSelect" class="form-select">

            <option value="en">English</option>

            <option value="hi">Hindi (हिन्दी)</option>

            <option value="ta">Tamil (தமிழ்)</option>

            <option value="te">Telugu (తెలుగు)</option>

            <option value="bn">Bengali (বাংলা)</option>

            <option value="mr">Marathi (मराठी)</option>

            <option value="gu">Gujarati (ગુજરાતી)</option>

            <option value="ml">Malayalam (മലയാളം)</option>

            <option value="kn">Kannada (ಕನ್ನಡ)</option>

            <option value="pa">Punjabi (ਪੰਜਾਬੀ)</option>

            <option value="ur">Urdu (اردو)</option>

            <option value="or">Odia (ଓଡ଼ିଆ)</option>

            <option value="as">Assamese (অসমীয়া)</option>

        </select>

        <input type="text" id="userText" class="form-control" placeholder="Type your message here...">

        <button class="btn" onclick="submitToPython()">Send</button>

    </div>

</div>

<script>

function submitToPython() {

    const msg = document.getElementById("userText").value;

    const lang = document.getElementById("langSelect").value;

    google.colab.kernel.invokeFunction('send\_message\_backend', [msg, lang], {});

    document.getElementById("userText").value = "";

}

</script>

"""

**# --- Chat Logic ---**

chat\_history = ""

chat\_display = widgets.HTML(value=f"<div id='chat'>{chat\_history}</div>")

def get\_timestamp():

    return datetime.now().strftime("%I:%M %p")

def send\_message\_backend(user\_message, selected\_language):

    global chat\_history

    user\_input = user\_message.strip().lower()

    timestamp = get\_timestamp()

    chat\_history += f"<p class='message user-message'><span class='icon'>👤</span><b>You:</b> {user\_message}<span class='timestamp'> {timestamp}</span></p>"

    chat\_display.value = f"<div id='chat'>{chat\_history}</div>"

    display(Javascript("autoScroll()"))

**# Typing indicator with animation**

    chat\_history += """

    <p class='message bot-message' id='typing'>

        <span class='icon'>🤖</span><b>Bot is typing</b>

        <span class='typing-dots'><span>.</span><span>.</span><span>.</span></span>

    </p>

    """

    chat\_display.value = f"<div id='chat'>{chat\_history}</div>"

    display(Javascript("autoScroll()"))

    if user\_input in pretrained\_responses:

        bot\_response = pretrained\_responses[user\_input]

    else:

        bot\_response = get\_ai\_response(user\_message)

    translated\_response = translate\_text(bot\_response, target\_lang=selected\_language)

    audio\_element = generate\_audio\_base64(translated\_response, lang=selected\_language)

    timestamp = get\_timestamp()

**# Remove typing indicator and add final response**

    chat\_history = chat\_history.replace(

        """<p class='message bot-message' id='typing'>

        <span class='icon'>🤖</span><b>Bot is typing</b>

        <span class='typing-dots'><span>.</span><span>.</span><span>.</span></span>

    </p>""",

      ""

    )

    chat\_history += f"<p class='message bot-message'><span class='icon'>🤖</span><b>Bot:</b> {translated\_response}{audio\_element}<span class='timestamp'> {timestamp}</span></p>"

    chat\_display.value = f"<div id='chat'>{chat\_history}</div>"

    display(Javascript("autoScroll()"))

from google.colab import output

output.register\_callback('send\_message\_backend', send\_message\_backend)

**# --- Render UI ---**

display(HTML(bootstrap\_style + auto\_scroll\_js))

display(HTML("<div id='container'>"))

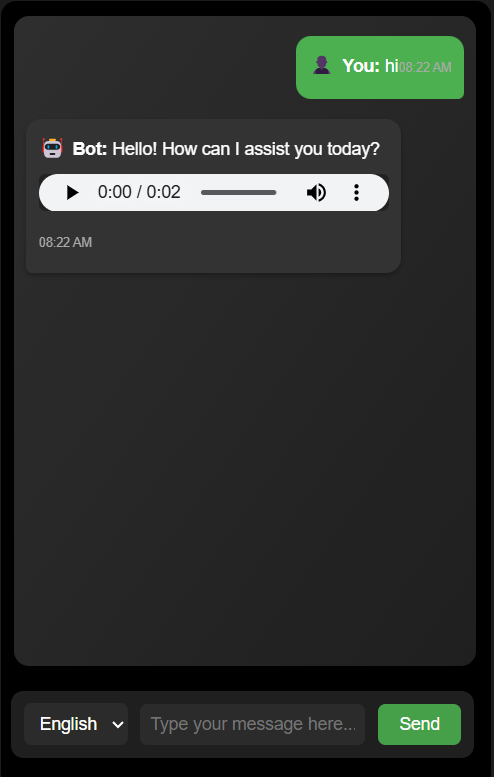
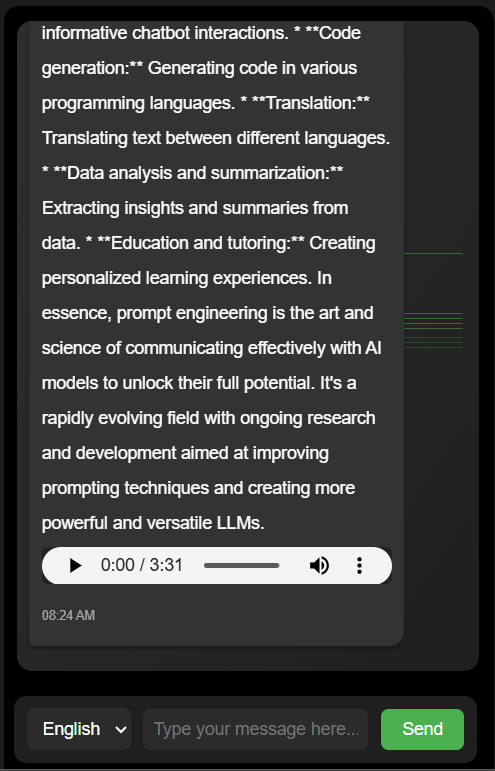
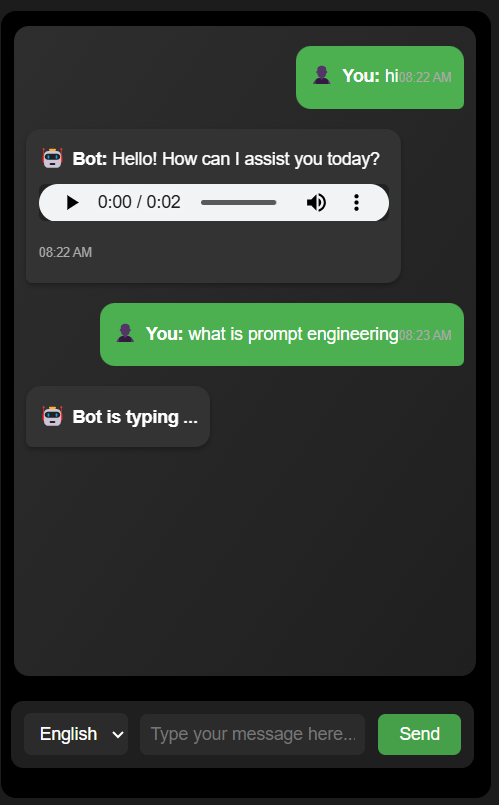
display(HTML("<div id='chat-container'>"))

display(chat\_display)

display(HTML(input\_ui\_html))

display(HTML("</div>"))  # end chat-container

display(HTML("</div> <br><br>"))  # end container

**OUTPUT SCREENS**